# Q1 Data processing

## 1. Tokenizer

本次作業我使用 BERT 及 RoBERTa 預訓練模型,並同時使用 其 tokenizer 實現 tokenize。BERT 的 tokenizer 將輸入文本拆分、 分詞,並添加特殊 tokens 做為輸入,如[CLS]為序列的開始、[SEP] 為分隔、[PAD]用以填充等等。而 RoBERTa 和 BERT 類似,但透 過更大的詞表提供性能。

## 2. Answer Span

- i. 透過 offset mapping, 能將 token 轉回原文本中字詞的位置, 並計算起始及結束位置。
- ii. 去除負長度及過長的答案,取出前 n\_best\_size,計算 softmax 並回傳機率最高者。

# Q2 Modeling with BERTs and their variants

### 1. Model 1

#### i. Model

bert-base-chinese

```
"_name_or_path": "bert-base-chinese",
                                            "_name_or_path": "bert-base-chinese",
"architectures": [
                                            "architectures": [
  "BertForMultipleChoice"
                                              "BertForQuestionAnswering"
"attention_probs_dropout_prob": 0.1,
                                            "attention_probs_dropout_prob": 0.1,
"classifier_dropout": null,
                                            "classifier_dropout": null,
"directionality": "bidi",
                                            "directionality": "bidi",
"hidden_act": "gelu",
                                            "hidden_act": "gelu",
"hidden_dropout_prob": 0.1,
                                            "hidden_dropout_prob": 0.1,
"hidden size": 768,
                                            "hidden_size": 768,
"initializer_range": 0.02,
                                           "initializer_range": 0.02,
"intermediate_size": 3072,
                                           "intermediate_size": 3072,
"layer_norm_eps": 1e-12,
                                            "layer_norm_eps": 1e-12,
"max_position_embeddings": 512,
                                            "max_position_embeddings": 512,
"model_type": "bert",
                                            "model_type": "bert",
"num_attention_heads": 12,
                                            "num_attention_heads": 12,
"num_hidden_layers": 12,
                                            "num_hidden_layers": 12,
"pad_token_id": 0,
                                            "pad_token_id": 0,
"pooler_fc_size": 768,
                                            "pooler_fc_size": 768,
"pooler_num_attention_heads": 12,
                                           "pooler_num_attention_heads": 12,
"pooler_num_fc_layers": 3,
                                            "pooler_num_fc_layers": 3,
"pooler_size_per_head": 128,
                                            "pooler_size_per_head": 128,
"pooler_type": "first_token_transform",
                                            "pooler_type": "first_token_transform",
"position_embedding_type": "absolute",
                                            "position_embedding_type": "absolute",
"torch_dtype": "float32",
                                            "torch_dtype": "float32",
                                            "transformers_version": "4.34.0",
"transformers_version": "4.34.0",
"type_vocab_size": 2,
                                            "type_vocab_size": 2,
                                            "use_cache": true,
"use_cache": true,
"vocab_size": 21128
                                            "vocab_size": 21128
```

#### ii. Performance

MC: 0.9498

QA: 76.936

### iii. Loss function

CrossEntropyLoss

## iv. Optimization algorithm, learning rate and batch size

AdamW()

Learning rate = 3e-5

Effective batch size (MC) = 2 \* 2

Effective batch size (QA) = 2 \* 2

#### 2. Model 2

#### i. Model

hfl/chinese-roberta-wwm-ext

```
"_name_or_path": "hfl/chinese-roberta-wwm-ext",
                                                             "_name_or_path": "hfl/chinese-roberta-wwm-ext
"architectures": [
                                                                "BertForQuestionAnswering"
   "BertForMultipleChoice"
                                                             "attention_probs_dropout_prob": 0.1,
"attention_probs_dropout_prob": 0.1,
                                                             "bos_token_id": 0,
"bos_token_id": 0,
                                                             "classifier_dropout": null,
"classifier dropout": null,
"directionality": "bidi",
"eos_token_id": 2,
                                                            "eos_token_id": 2,
"hidden_act": "gelu",
"hidden_act": "gelu",
"hidden_dropout_prob": 0.1,
                                                             "hidden_dropout_prob": 0.1,
"hidden_size": 768,
"initializer_range": 0.02,
                                                             "hidden_size": 768,
                                                             "initializer_range": 0.02,
                                                             "intermediate_size": 3072,
"intermediate_size": 3072,
                                                            "layer_norm_eps": 1e-12,
"layer_norm_eps": 1e-12,
                                                             "max_position_embeddings": 512,
"max_position_embeddings": 512,
                                                             "model_type": "bert'
"model_type": "bert",
                                                             "num_attention_heads": 12,
"num_attention_heads": 12,
                                                            "num_hidden_layers": 12,
"output_past": true,
"num_hidden_layers": 12,
"output_past": true,
"pad_token_id": 0,
                                                            "pad_token_id": 0,
"pooler_fc_size": 768,
                                                            "pooler_fc_size": 768,
"pooler_num_attention_heads": 12,
                                                             "pooler_num_attention_heads": 12,
"pooler_num_fc_layers": 3,
                                                            "pooler_size_per_head": 128,
"pooler_type": "first_token_transform",
"pooler_size_per_head": 128,
"pooler_type": "first_token_transform",
"position_embedding_type": "absolute",
                                                             "position_embedding_type": "absolute",
"torch_dtype": "float32",
"transformers_version": "4.34.0",
                                                             "torch_dtype": "float32
                                                            "transformers_version": "4.34.0",
"type vocab size": 2.
                                                             "type_vocab_size": 2,
                                                            "use_cache": true,
"vocab_size": 21128
"use_cache": true,
"vocab_size": 21128
```

#### ii. Performance

MC: 0.9598

QA: 81.555

#### iii. Loss function

CrossEntropyLoss

## iv. Optimization algorithm, learning rate and batch size

AdamW()

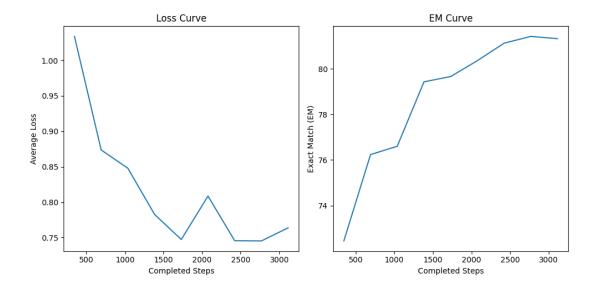
Learning rate = 5e-5 with warmup\_steps = 300

Effective batch size (MC) = 2 \* 2

Effective batch size (QA) = 16 \* 1

RoBERTa 相較 BERT 使用了更大的 BPE 詞彙表及更多的文本資料、更大的 batch size 進行預訓練,並使用動態 masking 機制且移除下一句預測任務。

# Q3 Curves (Validation set)



# Q4Pre-trained vs Not Pre-trained

## 1. Model

模型和 Model 2 一樣,但去除所有 pretrained weights。

```
"_name_or_path": "hfl/chinese-roberta-wwm-ext"
 "architectures": [
    "BertForQuestionAnswering"
],
"attention_probs_dropout_prob": 0.1,
"bos_token_id": 0,
"classifier_dropout": null,
"directionality": "bidi",
"eos_token_id": 2,
"hidden_act": "gelu",
"hidden_dropout_prob": 0.1,
"hidden_size": 768,
"initializer_range": 0.02,
"intermediate_size": 3072,
"Intermediate_size": 3072,
"layer_norm_eps": 1e-12,
"max_position_embeddings": 512,
"model_type": "bert",
"num_attention_heads": 12,
"num_hidden_layers": 12,
"output_past": true,
"pad_token_id": 0,
"pooler_fc_size": 768,
"pooler_num_attention_heads": 12,
"pooler_num_fc_layers": 3,
"pooler_size_per_head": 128,
"pooler_type": "first_token_transform",
"position_embedding_type": "absolute",
"torch_dtype": "float32",
"transformers_version": "4.34.0",
"type_vocab_size": 2,
"use_cache": true,
"vocab_size": 21128
```

## 2. Performance

i. Exact match: 4.653

ii. Kaggle score: 0.08589

可以看出預訓練及非預訓練模型在預測表現上的極大差別。

# Q5 Bonus

### 1. Model

severinsimmler/xlm-roberta-longformer-base-16384

```
{
    "_name_or_path": "severinsimmler/xlm-roberta-longformer-base-16384",
    "architectures": [
        "LongformerForQuestionAnswering"
],
    "attention_probs_dropout_prob": 0.1,
    "attention_window": [
        256,
        256,
        256,
        256,
        256,
        256,
        256,
        256,
        256,
        256,
        256,
        256,
        256,
        256,
        256,
        256,
        256,
        256,
        256,
        256,
        256,
        256,
        256,
        256,
        256,
        256,
        256,
        256,
        256,
        256,
        256,
        256,
        256,
        256,
        256,
        256,
        256,
        256,
        256,
        256,
        256,
        256,
        256,
        256,
        256,
        256,
        256,
        256,
        256,
        256,
        256,
        256,
        256,
        256,
        256,
        256,
        256,
        256,
        256,
        256,
        256,
        256,
        256,
        256,
        256,
        256,
        256,
        256,
        256,
        256,
        256,
        256,
        256,
        256,
        256,
        256,
        256,
        256,
        256,
        256,
        256,
        256,
        256,
        256,
        256,
        256,
        256,
        256,
        256,
        256,
        256,
        256,
        256,
        256,
        256,
        256,
        256,
        256,
        256,
        256,
        256,
        256,
        256,
        256,
        256,
        256,
        256,
        256,
        256,
        256,
        256,
        256,
        256,
        256,
        256,
        256,
        256,
        256,
        256,
        256,
        256,
        256,
        256,
        256,
        256,
        256,
        256,
        256,
        256,
        256,
        256,
        256,
        256,
        256,
        256,
        250,
        25082,

        "middle__strimple_distrimple_distrimple_
```

## 2. Performance

i. Exact match: 71.252

ii. Kaggle score: 0.6962

# 3. Loss function

CrossEntropyLoss

# 4. Optimization algorithm, learning rate and batch size

- i. AdamW()
- ii. Learning rate = 3e-5
- iii. Effective batch size = 4 \* 2